Cryptanalysis and Improvements of the Quasigroup Block Cipher

Matthew Battey\(^1\), Abhishek Parakh\(^2\) and William Mahoney\(^2\)

\(^1\)Principle Architect, Aspect Software
Computer Science Department
University of Nebraska at Omaha
mbattey@unomaha.edu

\(^2\)Nebraska University Center for Information Assurance
University of Nebraska at Omaha
aparakh, wmahoney@unomaha.edu

Abstract: This article presents results on the cryptanalysis of a quasigroup block cipher, which was previously proposed. The quasigroup block cipher provides an attractive encryption system for resource constrained environments. Here, we identify the “odd-bit” and “identical-word” problems with the cipher and recommend configurations of the QGBC to counter these. Following this analysis, we propose an improved variant of the QGBC, which doubles the block size, and quarters the total number of operations by per block.

Keywords: Quasigroup block cipher, cryptanalysis, algebraic analysis, linear analysis, known-plain-text attack, FPGA

I. Introduction

The evaluation results from NIST Statistical Test Suite [1] on the quasigroup block cipher (QGBC) proposed in [2, 3, 4] have shown good randomizing properties. Although this is an essential requirement for a block cipher to be secure, by no means this is a sufficient test for evaluating the “goodness” of the block cipher. As a result, we present linear and algebraic analysis of the proposed algorithm (which was first explored in [5]).

Over the last decade, due to the proliferation of network enabled devices, there has been a renewed interest in developing lightweight cryptosystems. Most cryptosystems are, however, targeted towards sensor networks or RFID applications, while not taking the unique demands of SCADA systems into consideration.

Supervisory Control and Data Acquisition (SCADA) systems are cyber-physical systems that are widely used to monitor and control automation processes. Examples include power plants, factories, water treatment plants, pipelines, and the energy grid. These systems are increasingly using new Internet Protocol (IP) based communication technologies for better integration with other enterprise information systems and accessibility from the Internet, which results in greater risk of attacks by unauthorized users. Attacks on SCADA systems can be more destructive than other cyber attacks because of the physical, real-world consequences they can have, such as loss of power, contamination of resources, and loss of machinery or even life.

Previously, attacking SCADA systems required physical proximity to the system. Now, however, these systems are increasingly monitored and controlled remotely over networks that are accessible from enterprise networks or in some cases even open to the public. This interconnectedness makes information more readily available to the operators or customers of the SCADA system, and facilitates easier management of the system. At the same time however, infrastructures that were once isolated are now accessible to the outside, with potentially devastating results. Many examples of critical infrastructure failures due to SCADA attacks exist in the literature and an extensive list can be found in Kolbe [6]. As a result these SCADA networks are at the fundamental foundations of our society and lifestyle, yet are difficult to secure, due in part to the complexity of their architectures. And the attacks are difficult to protect against for many reasons, not the least of which is the fact that SCADA systems are so different than traditional enterprise IT systems. As a result, some of the more mature security solutions in the enterprise do not carry over well into the SCADA infrastructure. For example, intrusion detection systems for SCADA are still new and attack patterns are still being developed and discovered. Secondly, SCADA systems are real-time systems, so encryption that significantly slows operation is not an option. Also, many of the devices used in this field are IP-enabled so they can be accessed over the Internet, and communicate over wireless networks. Remote desktop applications are also frequently used [7]. In addition, upgrading or replacing equipment with more up-to-date security is expensive. Finally, it can be difficult to detect whether an event is malicious or merely an accident. Security is often not given a high priority by those in charge of SCADA systems.

Quasigroup based systems can provide an advantage in SCADA systems (legacy and new) as they only require table look up operations and optionally modular arithmetic operations. In this paper we explore linear cryptanalysis of the QG-
BC and extend that analysis to performance improvements shown in a 256-bit variant. Here we will show the effect of altering parameters to the substitution function \( S() \) and permutation function \( P() \), and demonstrate the effect on correlations between a known plain text and cipher text. To this end, we will construct straw man QGBC systems, and use the results to uncover issues such as the “odd-bit” correlation problem. Then through algebraic cryptanalysis we will demonstrate issues, such as the “identical word” problem, that have arisen from previous versions of the QGBC; and then provide solutions to enhance the strength of the cryptosystem.

Our goal with these enhancements is to propose a version of the QGBC cryptosystem that not only maintains the statistical and analytical profile of previous work but reduces the number of clock cycles to implement. We are specifically targeting an FPGA infrastructure, as these devices allow for highly parallel calculation. Later in this paper we demonstrate an algorithm for a highly efficient FPGA implementation that can perform QGBC encryption with Cipher Block Chaining (CBC) in as few as 19 clock cycles, plus the cycles to exchange each block with the data bus.

### A. Previous Work on Lightweight Cryptosystems

Due to the increasing popularity of RFIDs and sensor networks several lightweight cryptographic algorithms have been developed in recent years [8, 9, 10, 11, 12]. The EU ECRYPT network began an organized effort to develop new stream ciphers for widespread adoption under the eSTREAM project [13]. The project was completed in 2008 and provided a number of stream ciphers that have high throughput and are especially suited for resource constrained hardware [14]. The project did not, however, look at development of block ciphers.

Many of the other lightweight cryptosystems have been modified (stripped down) versions of ECC [15, 16] and DES [17] algorithms. For example, [17] proposes a lightweight version of DES for RFID chips and uses a single s-box repeated eight times in order to improve on storage.

A PRINTcipher was proposed for integrated circuit printing [18] however its cryptanalysis has revealed some weaknesses [19, 20, 21]. Similarly Piccolo [22] was proposed as an ultra-lightweight block cipher for RFID tags and sensor nodes. Fault analysis of Piccolo is presented in [23, 24] where it is shown that the key candidates can be reduced significantly based on a few correct and faulty cipher texts.

TWINE was proposed [25] as a lightweight cipher for multiple platforms and low-end microcontrollers. Some cryptanalytic attacks have been proposed on it with slight improvements than brute force attacks [26].

PRESENT is another general purpose block cipher developed for low-power consumption and high chip efficiency [27]. It has also been adopted by International Organization for Standardization and International Electrotechnical Commission as a standard algorithm. LED is another compact block cipher [28] developed. Some weaknesses have been discovered in LED, as well, that enable attacks on it [29, 30].

These are only a few efforts related to development of lightweight cryptosystems, however their main applications have remained RFIDs and sensor networks and not specifically towards SCADA systems or CPS systems, that are often deployed in mission critical operations, where additional hardening against reverse engineering, cloning and side channel attacks may be required.

### II. Quasigroups

Quasigroups are mathematical constructs similar to groups. However, they do not necessarily possess the axioms of identity and associativity, but only require closure and inversion. In popular culture, we see quasigroups in the Sudoku game. In cryptography, quasigroups have been studied for at least 500 years, when Blaise de Vigenere constructed a polyalphabetic cipher that effectively performed a one time pad not unlike stream ciphers in use today (albeit trivial in modern terms) [31].

The quasigroup operation is a binary operation which we will denote using \( ; \) and \( \circ \) denotes its inverse. Also we will use \( N \) to denote the order of the quasigroup, or in other words the number of distinct elements within. Table 1 is an example of a quasigroup order \( N = 4 \) rendered as a Latin square. For a given row or column, each item appears only a single time; which alternately allows the quasigroup to be represented as a set of ordered triples, such as the following that describes the first row in tbl. 1: \( \{(0, 0, 2), (0, 1, 0), (0, 2, 3), (0, 3, 1), \ldots \} \). Membership in a quasigroup can also be defined by a known mathematical operation such as addition modulo \( N \) or bit-wise exclusive-or.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 1:** Sample quasigroup order \( N = 4 \)

Computation of the quasigroup operation may be performed by a look-up from the ordered triples set or Latin square. Say we wish to compute \( 2 \cdot 3 \) using the Latin square approach; we first look in row 2 then column 3 and find our answer is \( 2 \cdot 3 = 3 \) (the first row and column are the 0’th). To perform \( 1 \circ 1 \), we look in row 1 and find the column containing 1. We see that 1 is found in column 2, thus \( 1 \circ 1 = 2 \). To reduce computation, it is possible to produce inverse quasigroup, which lets one use the row and column indices as in the forward quasigroup.

#### A. Quasigroup Block Cipher

The QGBC consists of two primary functions \( S() \), substitution, and \( P() \), permutation. \( S() \) performs a polyalphabetic substitution based on the chosen quasigroup, while \( P() \) performs a bit-wise rotation of the entire block. This combination allows for every bit in the block to be computationaly chained to every other bit, while performing very simple operations. In (1), we see the definition of the QGBC cryptosystem as applied to a single block. For definitions including cipher block chaining, see previous work [4, 3].

Later in this paper we will refer to the block, which is all bits being processed, and the word, which is a regularly sized segment of the block. Word size is determined by the quasi-
then develop a strategy to attack the cipher as a whole.

Based on the exponential nature of the tabulation, the analyst must limit his/her scope to some portion of the cipher and formulate an overall approximation [32].

### III. Linear Cryptanalysis

Linear cryptanalysis examines relationships between plaintext and ciphertext, with the goal of determining affine approximations of the cipher [32]. First identified as a method to assess the FEAL cipher by Matsui and Yamagishi [33] and later applied to the DES cipher [34], linear cryptanalysis uses the probability of linear combinations of message and ciphertext to attack individual rounds and by extension the cipher as a whole.

When we apply a linear cryptanalysis, we compute the probability of every combination of input $M$ and output $C$ bits, for every possible input $M$ and key $K$. For simplicity, let us consider $n = |M| = |C|$ and $|K| = k$, which represent the size in bits of $M$, $C$, and $K$. The number of linear combinations from $M$ and $C$ is then $2^{2n} - 2^{n+1}$, while the number of inputs $M$ and $K$ is $2^{nk}$. Overall we see the complexity of exhaustively performing a linear analysis would be $O(2^{kn^2})$.

Based on the exponential nature of the tabulation, the analyst must limit his/her scope to some portion of the cipher and then develop a strategy to attack the cipher as a whole.

As a straw man let us consider an order 4 substitution matrix such as the one shown in:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

*Table 2: Substitution matrix with odd-bit bias*

If we consider $c_0 \oplus m_0 = 0$ (least order bits of $C$ and $M$) we see it has $p = \frac{1}{2}$; clearly demonstrating a bias on the odd bit. With this knowledge, the analyst can pass along probability to subsequent passes, tracking the actions on the bit and formulate an overall approximation [32].

### IV. Linear Cryptanalysis of QGBC

When we apply linear cryptanalysis to the QGBC cipher, we will use it to help identify minima for the cipher’s order. To this end, we construct linear combinations of plain text bits and cipher text bits. Although the past implementations of QGBC used 128 bit blocks with 256 bit keys, and quasigroups of order 256, the number of linear combinations is beyond the grasp of our available computational power, as this would require reviewing $(2^{256} - 2^{128})$ linear combinations for each of the $2^{184}$ results, or $(2^{140} - 2^{128})$ comparisons!

For this reason we will review straw man versions of the QGBC to review the effect of alterations to $S()$ and $P()$, by varying the quasigroup choice and rotational distance. We’ve used a shorthand to describe the quasigroup being evaluated.

In the following sections we will discuss seven experiments. The $1^{st}$ experiment (exp.) will be a quasigroup order 2, the $2^{nd}$ a quasigroup defined by addition modulo 4, the $3^{rd}$ and $4^{th}$ – quasigroups order 4 that have been randomized and shifted by two bits, the $5^{th}$ uses addition modulo 4 but rotations that are not two bits in distance, the $6^{th}$ uses the quasigroup from experiment 4 but shifts by distances other than two bits and finally in the $7^{th}$ experiment, we examine a single round of a quasigroup order 16.

#### A. Experiment 1: QG order 2

We begin the analysis with a divide and conquer approach [32]. Hence, first consider a trivial QGBC cipher, with the following conditions:

- $N = 2$
- $|M| = 4$
- Single round of the substitution function $S()$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

*Table 3: Quasigroup $N = 2$, similar to exclusive-or*

Since the quasigroup is of order $N = 2$, the size of each word is a single bit ($\frac{|M|}{\log_2 N} = 1$). We are able to perform an exhaustive examination, capturing all $K$, $M$ and $C$. Next we count the the linear combinations of plain text and ciphertext bits which “sum” to zero. We see that certain combinations have probability $p = 1$, while the remainder have probability $p = \frac{1}{2}$. The combinations with high probability are shown in (2).

- $\forall i, j \in (0, 1, 2, 3), i \neq j$;
- $c_i \oplus c_j \oplus m_i \oplus m_j = 0; p = 1$
- $c_0 \oplus c_1 \oplus c_2 \oplus c_3 \oplus m_0 \oplus m_1 \oplus m_2 \oplus m_3 = 0; p = 1$

(2) High probability linear combinations for $N = 2$ QGBC

The results show that if we consider a pair of any two input bits, with the corresponding output bits, these form a balanced linear combination, in all cases. From this we interpret this to be a trivially weak cryptosystem [32]. We will
investigate the cause of this later in the paper, via algebraic analysis.

B. Experiment 2: Addition Modulo 4, shift 2

Next, we will examine a QGBC system of order 4. Here each word is now represented with two bits. Consequently \(+ (mod) 4\) (denotes addition mod 4) is not identical to bitwise \(\oplus\) as was the case with \(N = 2\) QGBC.

- \(N = 4, |K| = 8, |M| = 8, w = 4\)
- \(P(C, 2)\) performs a left shift by a whole word (2 bits)
- 4 rounds of the \(S()\) and \(P()\) functions
- Quasigroup is defined by \(+ (mod N)\) seen in table 4

Through an exhaustive comparison of all \(K, M\) and \(C\) for this system, tabulations show that an “odd bit” problem appears. In every message word, there is an independent, 1:1 correlation \(p = 1\) of low order bits in the corresponding ciphertext word. The “odd bit” issue occurs because there is always an even number of additions applied to each word, causing the low-order bit to never fluctuate.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4: Quasigroup order \(N = 4\), defined by \(+ (mod N)\)

C. Experiment 3: Randomized QG ‘A’, shift 2

For this sampling, we again perform an exhaustive combination of all \(K, M,\) and \(C\), but this time, the quasigroup is not defined by \(+ (mod N)\) but by a randomized Latin square, which does not reduce to \(+ (mod N)\) [35, 36], and is defined by the Latin square specified in table 5.

- \(N = 4, |K| = 8, |M| = 8, w = 4\)
- \(P(C, 2)\) performs a left shift by a whole word (2 bits)
- 4 rounds of the \(S()\) and \(P()\) functions

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5: Quasigroup with poor linear analysis performance

We see that this configuration also fails linear analysis based on the high probability linear combinations, for brevity, three of which shown in (3).

\[
m_0 \oplus m_1 \oplus m_3 \oplus m_5 \oplus m_7 \oplus c_2 = 0 \\
m_1 \oplus m_2 \oplus m_4 \oplus m_6 \oplus c_2 \oplus c_3 = 0 \\
m_1 \oplus m_5 \oplus m_6 \oplus c_1 \oplus c_5 \oplus c_6 = 0 \\
\]

(3) \(p = 1\) Combinations for \(N = 4, QG \rightarrow +(mod N)\) randomized

D. Experiment 4: Randomized QG ‘B’, shift 2

Like Example A, Example B uses a randomized quasigroup that does not reduce to \(+ (mod n)\), but is defined by table 6.

- \(N = 4, |K| = 8, |M| = 8, w = 4\)
- \(P(C, 2)\) performs a left shift by a whole word (2 bits)
- 4 rounds of the \(S()\) and \(P()\) functions
- Quasigroup defined in tbl. 6

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6: Quasigroup with improved linear analysis performance

This configuration resulted in 24,954 linear combinations

of \(p = \frac{1}{2}\), a maximum \(p = \frac{5}{8}\) and a minimum \(p = \frac{3}{8}\)

and average \(p = 0.5 \pm 0.0058\).

E. Experiment 5: Addition modulo 4, shift \(\neq 2\)

In this configuration, we once again use a quasigroup equivalent to \(+ (mod N)\), but to counter the “odd-bit” problem, we perform the sequence \(S(), P(C, 1), S(), P(C, 3), S(), P(C, 3), S()\).

- \(N = 4, k = 8, |M| = 8, w = 4\)
- \(P(C, r), r = 1, 3, 3\) performs a left shift by 1 then 3 then 3 again
- 4 rounds of the \(S()\) and \(P()\) functions

This solution completely eradicated the “odd-bit” problem as well as removing all \(p = 1\) and/or \(p = 0\) linear combinations. Instead we find that 55,511 of \(2^{16}\) linear combinations have \(p = \frac{1}{2}\). Of the remainder the average probability is \(p = 0.5 \pm 0.0138\) with a maximum of \(p = 0.0688\) and 0.344.

F. Experiment 6: QG ‘B’, shift \(\neq 2\)

For this experiment we revisited the Example B Latin square, but this time used the 1-3-3 rotation pattern used in the previous example. The following are the configuration data:

- \(N = 4, k = 8, |M| = 8, w = 4\)
- \(P(C, r), r = 1, 3, 3\) performs a left shift by 1 then 3 then 3 again
- 4 rounds of the \(S()\) and \(P()\) functions

\(\) Quasigroup defined in tbl. 6
In this case we discovered 50,798 linear combinations had $p = \frac{1}{2}$, the minimum probability $p = \frac{3}{8}$ and a maximum probability of $p = 0.563$ and an average $p = 0.5 \pm 0.0101$. From these results we conclude, in conjunction with identification of the “odd bit” problem, we can see that $P()$ should not rotate on the word boundary.

G. Experiment 7: QG Addition modulo 16

In the final exhaustive linear analysis experiment, we evaluate an $+(\text{mod } N), N = 16$ quasigroup. Because of the limitation of processing capabilities, we use a block of 8 bits, but now with only two words.

- $N = 16, k = 8, |M| = 8, w = 2$
- Single round of $S()$

We see that a single linear combination $m_0 \oplus m_4 \oplus c_0 \oplus c_4 = 0$ has $p = 1$, i.e. suffers the “odd bit” problem. We can speculate that with $P()$ steps that rotate by less than a whole word will remove this issue. With this experiment concluding the linear analysis, let us now look to algebraic analysis of the QGBC.

V. Algebraic Cryptanalysis

Algebraic cryptanalysis allows us to examine the QGBC algorithm directly, identifying possible defects introduced by the mathematical interaction. We will examine the $N = 2$ quasigroup and also larger order quasigroups which have four words per block.

A. Algebraic Analysis QGBC $N = 2$

Algebraic analysis of the QGBC system can lead us to an understanding of the high probability of a linear combination found using exhaustive linear analysis. Let us first consider QGBC order 2. Here $S$ becomes a bitwise XOR or Not XOR, for simplicity we will consider:

$$v = w$$

$$S(C, M, K, i) :$$

$$c_0 = k_0 \oplus (m_1 \oplus k_{w-i-1})$$

$$\forall 1 \leq j < w : c_j = c_j - 1 \oplus (m_{j-1} \oplus m_j)$$

Previously, there was an assumption that $\cdot$ was neither associative nor distributive over $\oplus$. However, we know that $\oplus$ is associative. Thus if we expanded $S()$ we see:

$$k' = k_0 \oplus k_1 \oplus \ldots \oplus k_{n-1}$$

$$\forall 0 \leq j < w : c_j = k' \oplus m_j$$

This would indicate that we have introduced a single bit of randomness in the system, hence the system is trivially weak.

B. Algebraic Analysis QGBC $N > 2$

As long as a QGBC system is of order greater than 2 and the quasigroup does not reduce to a bitwise XOR group, we can assume that $\cdot$ is not distributive over $\oplus$. Thus, let us consider a worst case scenario, where there are 4 words in the key and 4 words in the block (this could represent four 64-bit words, for 256 bits in the key and block, or our four 2-bit words for an 8 bit block from before). To further review a worst-case, consider that each word in the plain text is the same and represented by the term $a$. With these conditions, the cipher collapses to the following:

$$\forall 0 \leq j < 4, m_j = a :$$

$$c_j = k_0 \cdot (k_3 \oplus a)$$

$$c_j' = k_1 \cdot (k_2 \oplus P(k_0 \cdot (k_3 \oplus a)))$$

$$c_j'' = k_2 \cdot (k_1 \oplus P(k_1 \cdot (k_2 \oplus P(k_0 \cdot (k_3 \oplus a))))$$

$$c_j''' = k_3 \cdot (k_0 \oplus P(k_2 \cdot (k_1 \oplus P(k_1 \cdot (k_2 \oplus P(k_0 \cdot (k_3 \oplus a))))))$$

(4) Expansion of the QGBC block cipher

Note: each additional apostrophe indicates an additional round of the cipher; since there are four words in the key, four rounds are applied. From this, we can infer some characteristics about our key. Specifically that the following hold true (see (5)), else an input of $a = 0$ would result in a trivially simple cipher.

$$k_2 \neq P(k_0 \cdot k_3)$$

$$k_1 \neq P(k_1 \cdot (k_2 \oplus P(k_0 \cdot k_3)))$$

$$k_0 \neq P(k_2 \cdot (k_1 \oplus P(k_1 \cdot (k_2 \oplus P(k_0 \cdot k_3))))$$

(5) Inequalities to strengthen QGBC

Further observation shows an “identical word” problem, which is to say when $M_i = M_j$ then $C_i = C_j$ also. This would be an obvious attack on the cipher, and point to a plain text attack based on such a construction. Thus the ability to attack the cipher would remain on the strength of the problem described by $c_j'''$.

VI. Improvement of the QGBC

If we make a change to the QGBC cipher we see that we can counter the “identical word” problem. Consider a construction such as shown in (6). With this configuration we see a single pass improvement as shown in (7).

$$S(C, M, K, i) :$$

$$c_0 = k_{(v-1 \mod v)} \oplus (k_{(i-v)} \cdot m_0)$$

$$\forall 1 \leq j < w : c_j = k_{(v-1-i \mod v)} \oplus (c_{j-1} \cdot m_j)$$

(6) Improved QGBC Substitution Function

The first round ($C_i'$) shown in (7) shows that we have eliminated the “identical word” problem, and additionally, we have introduced more of the key into each pass of the cipher. Reviewing the expression for $C_i''$, we see that we have introduced $2^k$ bits of the key (or their inverse) into the calculation, which in subsequent rounds affects every bit in the block. In fact, by round 2 ($C_i'''$) we have successfully integrated bits (or their inverse) from every word in the key into every word of the cipher text.
A. Application of the Improved QGBC

Although the improved QGBC substitution function 6 may be applied in general, a special case is particularly interesting, where the QGBC is implemented in Field Programmable Gate Array (FPGA) hardware. While some FPGAs possess memory components [37], buffers large enough to hold 64 KB [2, 4] typically require off-chip memory to keep the unit price under $5.00US [38]. The low-overhead QGBC (LO-QGBC) [3] provides an alternative, the number of clock cycles to implement the algorithm, as-is, is not competitive with AES in terms of clock-cycles per encrypted block [39, 40].

Instead, consider the improved QGBC in the 256 bit block configuration (QGBC-HP), with the following parameters:

$$
C'_0 = K_3 \oplus (K_0 \cdot A)
$$
$$
C'_1 = K_2 \oplus ((K_3 \oplus (K_0 \cdot A)) \cdot A)
$$
$$
C'_2 = K_1 \oplus ((K_2 \oplus ((K_3 \oplus (K_0 \cdot A)) \cdot A)) \cdot A)
$$
$$
C'_3 = K_0 \oplus ((K_1 \oplus ((K_2 \oplus ((K_3 \oplus (K_0 \cdot A)) \cdot A)) \cdot A)) \cdot A)
$$
$$
C''_0 = K_3 \oplus (K_1 \cdot C'_1)
= K_3 \oplus (K_1 \cdot (K_2 \oplus ((K_3 \oplus (K_0 \cdot A)) \cdot A)))
$$
...

(7) Single pass of improved QGBC w/ full-word rotation distance

Quasigroup Order \( N = 2^{64} \)
Quasigroup Definition Addition Modulo \( N \)
Key Size 256 bits
Block Size 256 bits
Words per Block 4
Number of Rounds 4

$$
S(C, M, K, i) :
$$
$$c_0 = k_{(v-1 \mod v)} \oplus (k_0 + m_0)
$$
$$\forall 1 \leq j < w : c_j = k_{(v-1-i \mod v)} \oplus (c_{j-1} + m_j)
$$

(8) QGBC-HP Substitution function

With this we can express \( S() \) as in (8). The entire cipher is pictured via block diagram in fig. 1, where \( P_{57} \) a 57 bit left rotation and \( P_{83} \) indicates an 83 bit left rotation. The first S block is expanded to show the internals, and each subsequent S block is identical.

With this specification, an FPGA solution requires a total of 512 bits of memory divided into two register banks. K (Key, 256 Bits) and A (Accumulator, 256 bits, receives the initialization block(IV) as well as Message text), a 64-Bit Adder, 64-Bit XOR, M-Bit XOR (for use in loading the Accumulator, assuming the data bus is M bits wide and less than 64 and is a divisor of 256), inversion and shift logic. Each round may be accomplished in 4 clock cycles (executing the XOR and Addition in a single clock) followed by a single clock to rotate bits. The core processing for the QGBC-HP algorithm can be performed in 19 clocks, plus the overhead of read-in/read-out. The algorithm for this process is show in Algorithm 1.

B. Experimental Evaluation of QGBC-HP

As with previous versions of the QGBC, the high performance variant has been implemented in software and evaluated with the NIST-STS test suite [1]. As recommended we encrypted the first 50 KB of Beowulf [41]; creating 1000 samples, each with a randomly generated key and IV. All of the samples were passed through the analysis suite and PASS/FAIL results were tabulated (see tbl. 7). Applying the recommended confidence interval defined by

$$
\hat{p} = m + \frac{3 \hat{p}(1-\hat{p})}{m}
$$

where \( \hat{p} = 1 - \alpha, \alpha = 0.01 \): the significance level, and \( m = 1000 \): the sample size; we see that the proportion of successes should be greater than 0.9805607. Based on this interval, we see that the QGBC-HP algorithm passed all of...
Battey et al.

**Data:** A : Accumulator Register 256 bits  
**Data:** K : Key Register 256 bits

```c
/* Load the Key */  
K ← data bus;  /* 256/M clocks */  
/* Load the IV */  
A ← data bus;  /* 256/M clocks */  
while Encrypting Do  
|  
| /* Perform the Cipher Block */  
| Chaining Step while loading the message text */  
| A = A ⊕ ← data bus;  /* 256/M clocks */  
| /* Execute 4 rounds of the block cipher */  
| A0 = K3 ⊕ (K0 + A0);  /* 1 clock */  
| A1 = K2 ⊕ (A0 + A1);  /* 1 clock */  
| A2 = K1 ⊕ (A1 + A2);  /* 1 clock */  
| A3 = K0 ⊕ (A2 + A3);  /* 1 clock */  
for i = 1 to 3 Do  
| if i = 1 then  
| left rotate A by 57;  /* 1 clock */  
| else  
| left rotate A by 83;  /* 1 clock */  
| end  
end  
A = A ⊕ ← data bus;  /* 256/M clocks */  
```

**Algorithm 1:** QGBC-HP FPGA Algorithm

```c
/* Execute 4 rounds of the block cipher */  
A0 = K3 ⊕ (K0 + A0);  /* 1 clock */  
A1 = K2 ⊕ (A0 + A1);  /* 1 clock */  
A2 = K1 ⊕ (A1 + A2);  /* 1 clock */  
A3 = K0 ⊕ (A2 + A3);  /* 1 clock */  
end  
```

**Table 7:** QGBC-HP NIST-STS Results

<table>
<thead>
<tr>
<th>NIST-STS Test</th>
<th>Success Rate</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approximate Entropy</td>
<td>980/1000</td>
<td>PASS</td>
</tr>
<tr>
<td>Block Frequency</td>
<td>991/1000</td>
<td>PASS</td>
</tr>
<tr>
<td>Cumulative Sums-Forward</td>
<td>994/1000</td>
<td>PASS</td>
</tr>
<tr>
<td>Cumulative Sums-Reverse</td>
<td>998/1000</td>
<td>PASS</td>
</tr>
<tr>
<td>FFT</td>
<td>997/1000</td>
<td>PASS</td>
</tr>
<tr>
<td>Frequency</td>
<td>994/1000</td>
<td>PASS</td>
</tr>
<tr>
<td>Longest Run</td>
<td>997/1000</td>
<td>PASS</td>
</tr>
<tr>
<td>Rank</td>
<td>997/1000</td>
<td>PASS</td>
</tr>
<tr>
<td>Runs</td>
<td>990/1000</td>
<td>PASS</td>
</tr>
<tr>
<td>Serial 1</td>
<td>992/1000</td>
<td>PASS</td>
</tr>
<tr>
<td>Serial 2</td>
<td>988/1000</td>
<td>PASS</td>
</tr>
</tbody>
</table>

**VIII. Conclusions**

With this examination, we have found that the choice of the quasigroup influences the strength of the cryptosystem. Also, we have shown the permutation function $P()$ must not rotate by a distance equal to $\log_2 N$, else we encounter the “odd-bit” problem. Finally, through algebraic analysis we have seen that previous proposals for the QGBC should be amended, and have provided a solution that not only counters the “identical word” but introduces more of the key into each pass of the cipher.

As we continue to investigate quasigroup based block ciphers, we hope to gain a better understanding of performance in both cryptographic quality and computational efficiency. We have proposed the QGBC-HP variant which is particularly well suited to low cost FPGA applications, as well as offering improvements in 64-bit CPUs. We have shown that the QGBC-HP not only passes algebraic examination, but also passes standardized statistical testing. Further examination of the QGBC is still warranted. Inspection using techniques such as differential cryptanalysis or integral cryptanalysis may shed new light on the strength of the system. Also, physical FPGA implementations and operational comparisons with existing systems will shed light on
the relative throughput of the QGBC system. Research such as these will provide use with better understanding of QGBC, providing further insight and confidence in the system.

References


Author Biographies

Matthew Battey is a Principal Solution Architect with Aspect Software. Novel applications of coding theory led him to expand upon his more than 22 years of industry experience and focus on Information Assurance at the University of Nebraska at Omaha. His interests lie in identification of best-practices in all arenas of computer security, cryptography and network protocols. He also maintains a blog on software development: http://battey-muse.blogspot.com.

Abhishek Parakh is from Jodhpur, Rajasthan, India. He received his M.S. in Electrical Engineering from Louisiana State University, Baton Rouge, LA and his Ph.D. in Computer Science from Oklahoma State University, Stillwater, OK. He is currently an Assistant Professor of Information Assurance at University of Nebraska, Omaha and a faculty member of Nebraska University Center for Information Assurance. His research interests lie in cryptography and network security.

William Mahoney received his B.A. and B.S. degrees from Southern Illinois University, and his M.A. and Ph.D. degrees from the University of Nebraska. He is an Associate Professor in the College of Information Science and Technology, University of Nebraska at Omaha, and is the Director of the Nebraska University Center for Information Assurance (NU-CIA). His primary research interests include language compilers, hardware and instruction set design, and code generation and optimization, as these topics relate to information assurance goals. As such is interests are in areas such as code obfuscation, reverse engineering and anti-reverse engineering techniques, and vulnerability analysis. Industrial control systems are a specific target of these research areas. Prior to the Kiewit Institute Dr. Mahoney worked for 20+ years in the computer design industry, specifically in the areas of embedded computing and real-time operating systems. During this time he was also on the part time faculty of the University of Nebraska at Omaha. He regularly teaches in both the Information Assurance and Computer Science areas and is a reviewer for several Information Warfare publications and conferences.